UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/799,351	03/12/2004	David Dehghan	MSFT122464	6346

26389 7590 12/30/2008

CHRISTENSEN, O'CONNOR, JOHNSON, KINDNESS, PLLC
1420 FIFTH AVENUE
SUITE 2800
SEATTLE, WA 98101-2347

EXAMINER	
LE, MIRANDA	

ART UNIT	PAPER NUMBER
2169	

MAIL DATE	DELIVERY MODE
12/30/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/799,351	DEHGHAN ET AL.	
	Examiner	Art Unit	
	MIRANDA LE	2169	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *12 August 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-17 and 30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-17 and 30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *08/12/08*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Claims 1-17, 30 are pending in this application. This action is made Final.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

Claims 1-3, 10, 11, 17, 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mayer et al. (US Pub No. 20020019864), in view of Melchione et al. (US Pub No. 20030200300), and further in view of Crudele et al. (US Patent No. 6,973,647).

As per claim 1, Mayer teaches a software update distribution system *(i.e. FIG. 9 is a block diagram showing an order management for distributing software within an IT environment, in accordance with the invention, [0046])* for distributing *(i.e. Once all the changes are verified, the automatic execution of the changes is started that provides an automatic updating of the respective resources of the managed IT environment, [0084]; Once the administrator is satisfied with the changes, the changes are confirmed and updated within the change request, [0083])* a software update over a communication network for distribution to client computers, comprising:

a root *(i.e. IT devices 31, Fig. 3)* update service node *(i.e. FIG. 1 is an hierarchical arrangement of distributed agents and managers according to the invention, [0038])*; and

a plurality of child update service nodes *(i.e. The following FIG. 3 which is a schematic view of an IT environment 30 comprising IT devices 31, 31', 31", 31'" serves to illustrate the principle of managed elements 32, 32", 32'". ITCM models a managed system by elements. A system is nothing else than a container for parameterized elements. The concrete parameter values describe the system and its component, [0067])*;

wherein the root update service node and the at least one child update service node are organized in a hierarchical manner *(i.e. FIG. 1 is an hierarchical arrangement of distributed agents and managers according to the invention, [0038])* such that the root update service node is a parent update service node to at least one child update service node *(i.e. IT devices 31', 31", 31'", Fig. 3)*,

wherein each update service node, except the root update service node *(See Figs. 1, 3)*, has a parent update service node *(i.e. IT devices 31' is parent of IT devices 31", See Fig. 3)*, wherein each of the plurality of child update service nodes is configured to operate as a parent update service node to another child update service node, and wherein at least one child update service node of the plurality of child update service nodes is a parent update service node to another child update service node of the plurality of child update service nodes *(i.e. Each agent advantageously can have a local management storage that holds the information about the current configuration of the managed elements in a data processing resource e.g. a computer system which is assigned to that agent, [0027]; The agent means can perform a delta detection based on the managed elements that represent the smallest managed unit under the concept according to the invention. It is emphasized hereby that typical managed elements are an entire software package, a printer or a user. But the managed elements are not used to represent individual files. It is emphasized that the data processing resources, accordingly, can be any software, hardware, data files, users or user profiles, i.e. all data processing elements which can principally be managed, [0028]*); and

wherein the root update service node obtains a software update from a software provider *(i.e. It is noteworthy that the IT devices 4, in principle, can also represent software (IT) resources like Web Browsers or any other intercommunication software which are interconnected via large scale networks like the Internet or proprietary intranets, [0053])*, and wherein each of the at least

one child update service nodes obtains the software update for distribution *(i.e. FIG. 9 is a block diagram showing an order management for distributing software within an IT environment, in accordance with the invention, [0046])* to client computers by obtaining the software update from its parent software update node *(i.e. Each agent advantageously can have a local management storage that holds the information about the current configuration of the managed elements in a data processing resource e.g. a computer system which is assigned to that agent, [0027]; The agent means can perform a delta detection based on the managed elements that represent the smallest managed unit under the concept according to the invention. It is emphasized hereby that typical managed elements are an entire software package, a printer or a user. But the managed elements are not used to represent individual files. It is emphasized that the data processing resources, accordingly, can be any software, hardware, data files, users or user profiles, i.e. all data processing elements which can principally be managed, [0028])*.

includes an administration application programming interface (API) *(i.e. Once the element types are defined the administrator can combine them to configuration sets. These sets are used to define pre-configured sub-systems that can later be used to configure a system. The relation between element types and configuration sets will be described later referring to FIG. 8, [0094])*.

Mayer does not explicitly teach:

includes an administration application programming interface (API) through which an administrator established a set of rules for distributing software updates from the update service node to its child update service nodes.

Melchione teaches an administration application programming interface (API) through which an administrator established rules for distributing software updates from the update service node to its child update service nodes *(i.e. In one case, administered software may be downloaded and run at a customer computer. In other cases, administered software represents some portion of administered software in processing communication with other administered software residing on other customer computers or at the provider. In another case, administered software may run at the provider with only a thin client (e.g., a web browser) running at a customer computer (e.g., displaying administered software output). All these cases, represent examples of provider-hosted application services, [0020]; In certain described examples, the software being administered is anti-virus software. New releases of anti-virus software can be automatically provided according to the software administration directives set by a customer administrator. In some case, administered software includes an agent that polls the data center to obtain administered software updates, [0021])*.

It would have been obvious to one of ordinary skill of the art having the teaching of Mayer and Melchione at the time the invention was made to modify the system of Mayer to include the limitations as taught by Melchione. One of ordinary skill in the art would be motivated to make this combination in order to represent some portion of administered software in processing communication

with other administered software residing on other customer computers or at the provider in view of Melchione, as doing so would give the added benefit of enabling new releases of anti-virus software to be automatically provided according to the software administration directives set by a customer administrator as taught by Melchione ([0021]).

Mayer, Melchione do not explicitly teach a set of rules.

Crudele teaches a set of rules *(i.e. a software deployment tool cooperable with a software package including a software package file incorporating at least one action defining respective modifications to said client processing system and at least one file required to implement said at least one modifying action, said tool comprising: a plurality of classes, each class corresponding to a respective type of action; means for reading said software package file and instantiating a class having attributes corresponding to the respective type of each of the at least one action of said software package file and setting the attributes of the at least one class according to the respective action definition in said software package file, means for executing a check method on at least one of each of said at least one class instances to determine if a deployment operation can be implemented in a specified first mode; means, responsive to a successful check, for executing a method on each of said at least one class instances in said first mode; and means, responsive to check failure of any class instance, for executing a method on each of said at least one class instances in a second less preferable mode, col. 2, lines 10-34).*

It would have been obvious to one of ordinary skill of the art having the teaching of Mayer, Melchione and Crudele at the time the invention was made to modify the system of Mayer, Melchione to include the limitations as taught by Crudele. One of ordinary skill in the art would be motivated to make this combination in order to provide a complete definition of the actions involved in a software distribution in view of Crudele (Abstract), as doing so would give the added benefit of satisfying the need when a software package is available to a target endpoint, an engine resident on the target can be instructed via the management agent to decode the software package from the file into memory and then to perform various software distribution operations including installing, removing and modifying the software and configuration of the endpoint as taught by Crudele (Abstract).

As per claim 11, Mayer teaches an update service node for distributing software *(i.e. FIG. 9 is a block diagram showing an order management for distributing software within an IT environment, in accordance with the invention, [0046])* updates to client computers and to child update service nodes *(i.e. FIG. 9 is a block diagram showing an order management for distributing software within an IT environment, in accordance with the invention, [0046])*, wherein the update service node is organized in a hierarchy *(i.e. FIG. 1 is an hierarchical arrangement of distributed agents and managers according to the invention, [0038])* of a plurality of similarly configured update service nodes *(i.e. Once all the changes are verified, the automatic execution of the changes is started that*

provides an automatic updating of the respective resources of the managed IT environment, [0084]; Once the administrator is satisfied with the changes, the changes are confirmed and updated within the change request, [0083]), the update service node comprising:

an update store *(i.e. IT devices 31, Fig. 3)* for storing software updates *(i.e. Typical managed elements 12 are an entire (installed) software package, [0057])*;

an update web service through which the update service node obtains software updates from a parent update service node *(i.e. IT devices 31, Fig. 3)* over a communication network, and through which the update service node distributes software updates to child update service nodes *(i.e. IT devices 31' is parent of IT devices 31", See Fig. 3)* in the hierarchy *(See Figs. 1, 3)* over the communication network *(i.e. Each agent advantageously can have a local management storage that holds the information about the current configuration of the managed elements in a data processing resource e.g. a computer system which is assigned to that agent, [0027]; The agent means can perform a delta detection based on the managed elements that represent the smallest managed unit under the concept according to the invention. It is emphasized hereby that typical managed elements are an entire software package, a printer or a user. But the managed elements are not used to represent individual files. It is emphasized that the data processing resources, accordingly, can be any software, hardware, data files, users or user profiles, i.e. all data processing elements which can principally be managed, [0028])*;

an administration application programming interface (API) *(i.e. Once the element types are defined the administrator can combine them to configuration sets. These sets are used to define pre-configured sub-systems that can later be used to configure a system. The relation between element types and configuration sets will be described later referring to FIG. 8, [0094])*.

a child update module for determining which software updates are available to be distributed to its child update service nodes *(i.e. Utilizing intelligent agents implies a high scalability of the entire management solution. Each agent can work independent from each other. This allows spreading the work down to all agents and letting them determine the required actions by making a desired--current analysis and initiate the actions. For example, software is installed by telling the agent that the software must be available and installed on the system. The agent will check whether the software is already installed on the system or whether it has to be installed. It will organize a copy of the software by asking its parent agent in the IT environment hierarchy to get the software. Once a node up in the hierarchy has the software, the software will be recursively copied down the hierarchy. Once the software is available, the agent will initiate the installation process of the software on the system. The above procedure is illustrated in detail referring to FIG. 5 which shows only a part of the entire IT network depicted in FIG. 1, [0085])*.

Mayer does not specifically teach:

includes an administration application programming interface (API) through which an administrator established a set of rules for distributing software updates from the update service node to its child update service nodes.

Melchione teaches an administration application programming interface (API) through which an administrator established rules for distributing software updates from the update service node to its child update service nodes *(i.e. In one case, administered software may be downloaded and run at a customer computer. In other cases, administered software represents some portion of administered software in processing communication with other administered software residing on other customer computers or at the provider. In another case, administered software may run at the provider with only a thin client (e.g., a web browser) running at a customer computer (e.g., displaying administered software output). All these cases, represent examples of provider-hosted application services, [0020]; In certain described examples, the software being administered is anti-virus software. New releases of anti-virus software can be automatically provided according to the software administration directives set by a customer administrator. In some case, administered software includes an agent that polls the data center to obtain administered software updates, [0021])*.

It would have been obvious to one of ordinary skill of the art having the teaching of Mayer and Melchione at the time the invention was made to modify the system of Mayer to include the limitations as taught by Melchione. One of ordinary skill in the art would be motivated to make this combination in order to represent some portion of administered software in processing communication

with other administered software residing on other customer computers or at the provider in view of Melchione, as doing so would give the added benefit of enabling new releases of anti-virus software to be automatically provided according to the software administration directives set by a customer administrator as taught by Melchione ([0021]).

Mayer, Melchione do not explicitly teach a set of rules.

Crudele teaches a set of rules *(i.e there is provided a software deployment tool cooperable with a software package including a software package file incorporating at least one action defining respective modifications to said client processing system and at least one file required to implement said at least one modifying action, said tool comprising: a plurality of classes, each class corresponding to a respective type of action; means for reading said software package file and instantiating a class having attributes corresponding to the respective type of each of the at least one action of said software package file and setting the attributes of the at least one class according to the respective action definition in said software package file, means for executing a check method on at least one of each of said at least one class instances to determine if a deployment operation can be implemented in a specified first mode; means, responsive to a successful check, for executing a method on each of said at least one class instances in said first mode; and means, responsive to check failure of any class instance, for executing a method on each of said at least one class instances in a second less preferable mode, col. 2, lines 10-34).*

It would have been obvious to one of ordinary skill of the art having the teaching of Mayer, Melchione and Crudele at the time the invention was made to modify the system of Mayer, Melchione to include the limitations as taught by Crudele. One of ordinary skill in the art would be motivated to make this combination in order to provide a complete definition of the actions involved in a software distribution in view of Crudele (Abstract), as doing so would give the added benefit of satisfying the need when a software package is available to a target endpoint, an engine resident on the target can be instructed via the management agent to decode the software package from the file into memory and then to perform various software distribution operations including installing, removing and modifying the software and configuration of the endpoint as taught by Crudele (Abstract).

As per claim 30, Mayer teaches a method for facilitating the distribution *(i.e. Once all the changes are verified, the automatic execution of the changes is started that provides an automatic updating of the respective resources of the managed IT environment, [0084]; Once the administrator is satisfied with the changes, the changes are confirmed and updated within the change request, [0083])* of software updates *(i.e. elements are entire software, [0028])* in a hierarchical arrangement *(i.e. FIG. 1 is an hierarchical arrangement of distributed agents and managers according to the invention, [0038])*, the method comprising:

providing a root update service node *(i.e. IT devices 31, Fig. 3)* of a hierarchy of update services nodes *(i.e. The following FIG. 3 which is a*

schematic view of an IT environment 30 comprising IT devices 31, 31', 31", 31'" serves to illustrate the principle of managed elements 32, 32", 32'". ITCM models a managed system by elements. A system is nothing else than a container for parameterized elements. The concrete parameter values describe the system and its component, [0067]);

providing a plurality of child update service nodes *(i.e. IT devices 31', 31", 31'", Fig. 3)* organized in a hierarchy under the root update service node *(See Figs. 1, 3)*; and

providing software *(i.e. elements are entire software, [0028])* update information *(i.e. the information about the current configuration of the managed elements, [0027])* corresponding to software updates *(i.e. Once all the changes are verified, the automatic execution of the changes is started that provides an automatic updating of the respective resources of the managed IT environment, [0084]; Once the administrator is satisfied with the changes, the changes are confirmed and updated within the change request, [0083])* for distribution to client computers connected to update services nodes in the hierarchy of update service nodes *(i.e. Each agent advantageously can have a local management storage that holds the information about the current configuration of the managed elements in a data processing resource e.g. a computer system which is assigned to that agent, [0027]; The agent means can perform a delta detection based on the managed elements that represent the smallest managed unit under the concept according to the invention. It is emphasized hereby that typical managed elements are an entire software package, a printer or a user. But the*

managed elements are not used to represent individual files. It is emphasized that the data processing resources, accordingly, can be any software, hardware, data files, users or user profiles, i.e. all data processing elements which can principally be managed, [0028]);

wherein the root update service node is a parent update service node to at least one of the plurality of child update service nodes *(i.e. IT devices 31', 31", 31"', Fig. 3)* and wherein each update service node, except the root update service node, has a parent update service node *(i.e. IT devices 31' is parent of IT devices 31", See Fig. 3)*;

wherein each of the plurality of child update service nodes is configured to operate as a parent update service node *(i.e. Once all the changes are verified, the automatic execution of the changes is started that provides an automatic updating of the respective resources of the managed IT environment, [0084]; Once the administrator is satisfied with the changes, the changes are confirmed and updated within the change request, [0083])* to another child update service node and at least one child update service node of the plurality of child update service nodes is a parent update service node to another child update service node of the plurality of child update service nodes *(i.e. Each agent advantageously can have a local management storage that holds the information about the current configuration of the managed elements in a data processing resource e.g. a computer system which is assigned to that agent, [0027]; The agent means can perform a delta detection based on the managed elements that represent the smallest managed unit under the concept according to the*

invention. It is emphasized hereby that typical managed elements are an entire software package, a printer or a user. But the managed elements are not used to represent individual files. It is emphasized that the data processing resources, accordingly, can be any software, hardware, data files, users or user profiles, i.e. all data processing elements which can principally be managed, [0028]).

includes an administration application programming interface (API) *(i.e. Once the element types are defined the administrator can combine them to configuration sets. These sets are used to define pre-configured sub-systems that can later be used to configure a system. The relation between element types and configuration sets will be described later referring to FIG. 8, [0094]).*

Mayer does not specifically teach:

includes an administration application programming interface (API) through which an administrator established a set of rules for distributing software updates from the update service node to its child update service nodes.

Melchione teaches an administration application programming interface (API) through which an administrator established rules for distributing software updates from the update service node to its child update service nodes *(i.e. In one case, administered software may be downloaded and run at a customer computer. In other cases, administered software represents some portion of administered software in processing communication with other administered software residing on other customer computers or at the provider. In another case, administered software may run at the provider with only a thin client (e.g., a web browser) running at a customer computer (e.g., displaying administered*

software output). All these cases, represent examples of provider-hosted application services, [0020]; In certain described examples, the software being administered is anti-virus software. New releases of anti-virus software can be automatically provided according to the software administration directives set by a customer administrator. In some case, administered software includes an agent that polls the data center to obtain administered software updates, [0021]).

It would have been obvious to one of ordinary skill of the art having the teaching of Mayer and Melchione at the time the invention was made to modify the system of Mayer to include the limitations as taught by Melchione. One of ordinary skill in the art would be motivated to make this combination in order to represent some portion of administered software in processing communication with other administered software residing on other customer computers or at the provider in view of Melchione, as doing so would give the added benefit of new releases of enabling anti-virus software to be automatically provided according to the software administration directives set by a customer administrator as taught by Melchione ([0021]).

Mayer, Melchione do not explicitly teach a set of rules.

Crudele teaches a set of rules *(i.e. there is provided a software deployment tool cooperable with a software package including a software package file incorporating at least one action defining respective modifications to said client processing system and at least one file required to implement said at least one modifying action, said tool comprising: a plurality of classes, each class corresponding to a respective type of action; means for reading said software*

package file and instantiating a class having attributes corresponding to the respective type of each of the at least one action of said software package file and setting the attributes of the at least one class according to the respective action definition in said software package file, means for executing a check method on at least one of each of said at least one class instances to determine if a deployment operation can be implemented in a specified first mode; means, responsive to a successful check, for executing a method on each of said at least one class instances in said first mode; and means, responsive to check failure of any class instance, for executing a method on each of said at least one class instances in a second less preferable mode, col. 2, lines 10-34).

It would have been obvious to one of ordinary skill of the art having the teaching of Mayer, Melchione and Crudele at the time the invention was made to modify the system of Mayer, Melchione to include the limitations as taught by Crudele. One of ordinary skill in the art would be motivated to make this combination in order to provide a complete definition of the actions involved in a software distribution in view of Crudele (Abstract), as doing so would give the added benefit of satisfying the need when a software package is available to a target endpoint, an engine resident on the target can be instructed via the management agent to decode the software package from the file into memory and then to perform various software distribution operations including installing, removing and modifying the software and configuration of the endpoint as taught by Crudele (Abstract).

As per claim 2, Mayer teaches the software update distribution system of claim 1, wherein the root update service node comprises:

an update store for storing software updates *(i.e. Typical managed elements 12 are an entire (installed) software package, [0057])*;

an update web service through which the root update service node distribute software updates to its child update service nodes over the communication network *(i.e. FIG. 9 is a block diagram showing an order management for distributing software within an IT environment, in accordance with the invention, [0046])*; and

a software provider interface through which a software provider submits its software update over the communication network to the root update distribution node *(i.e. It is noteworthy that the IT devices 4, in principle, can also represent software (IT) resources like Web Browsers or any other intercommunication software which are interconnected via large scale networks like the Internet or proprietary intranets, [0053])*.

As per claim 3, Mayer teaches the software update distribution system of claim 2, wherein each of the plurality of child update service nodes comprises:

an update store for storing software updates *(i.e. Typical managed elements 12 are an entire (installed) software package, [0057])*;

an update web service through which the child update service node obtains software updates form its parent update service node over the communication network, and through which the child update service node

distributes software updates to its update service nodes over the communication network *(i.e. FIG. 9 is a block diagram showing an order management for distributing software within an IT environment, in accordance with the invention, [0046])*;

a child update module for determining which software updates are available to be distributed to its child update service nodes *(i.e. Utilizing intelligent agents implies a high scalability of the entire management solution. Each agent can work independent from each other. This allows spreading the work down to all agents and letting them determine the required actions by making a desired--current analysis and initiate the actions. For example, software is installed by telling the agent that the software must be available and installed on the system. The agent will check whether the software is already installed on the system or whether it has to be installed. It will organize a copy of the software by asking its parent agent in the IT environment hierarchy to get the software. Once a node up in the hierarchy has the software, the software will be recursively copied down the hierarchy. Once the software is available, the agent will initiate the installation process of the software on the system. The above procedure is illustrated in detail referring to FIG. 5 which shows only a part of the entire IT network depicted in FIG. 1, [0085])*.

Crudele teaches a set of rules *(i.e. According to the present invention there is provided a software deployment tool cooperable with a software package including a software package file incorporating at least one action defining*

respective modifications to said client processing system and at least one file required to implement said at least one modifying action, col. 2, lines 10-34).

As to claims 10, 17, Melchione teaches the software update distribution system of claim 3, wherein the child update service node may be selectively configured to periodically obtain available software updates from the parent update service node *(i.e. The nodes can include agent software that periodically communicates with the data center 712, [0078]).*

5. Claims 4-9, 12-16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mayer et al. (US Pub No. 20020019864), in view of Melchione et al. (US Pub No. 20030200300), in view of Crudele et al. (US Patent No. 6,973,647), and further in view of Donohue et al. (US Patent No. 6,199,204).

As to claims 4, 12, Mayer teaches the software update distribution system of claim 3, wherein each of the plurality of child update service nodes further comprises:

a reporting module for generating and sending update activity reports to the parent update service node *(i.e. . It will organize a copy of the software by asking its parent agent in the IT environment hierarchy to get the software, [0085])*; and

a client update module for distributing software updates to client computers *(i.e. FIG. 9 is a block diagram showing an order management for*

distributing software within an IT environment, in accordance with the invention, [0046]).

Mayer, Melchione, Crudele do not specifically teach:

an authentication and authorization module for determining whether an update service node is authorized to obtain software updates from the child update service node.

Donohue teaches:

an authentication and authorization module for determining whether an update service node is authorized to obtain software updates from the child update service node *(i.e. The updater component preferably also includes a mechanism for verifying the authenticity of downloaded software, using cryptographic algorithms. This avoids the need for dedicated, password-protected or otherwise protected software resource repository sites. The software resources can be anywhere on the network as long as they are correctly named and or posted to the network search engines, col. 5, lines 36-42)*.

It would have been obvious to one of ordinary skill of the art having the teaching of Mayer, Melchione, Crudele, Donohue at the time the invention was made to modify the system of Mayer, Melchione, Crudele to include the limitations as taught by Donohue. One of ordinary skill in the art would be motivated to make this combination in order to verify the authenticity of downloaded software in view of Donohue (col. 5, lines 36-42), as doing so would give the added benefit of efficiently protecting software resource repository sites as taught by Donohue (col. 5, lines 36-42).

As to claims 5, 13, Mayer teaches the update store comprises an update content store in which the update payload for the software update is stored, and an update information store in which update metadata for the software update is stored *(i.e. . It will organize a copy of the software by asking its parent agent in the IT environment hierarchy to get the software, [0085])*.

As to claims 6, 14, Mayer teaches the child update service node obtains the software update from the parent update service by obtaining update metadata for the software update form the parent update service node, and separately obtaining the update payload for the software update from the parent update service node *(i.e. It will organize a copy of the software by asking its parent agent in the IT environment hierarchy to get the software. Once a node up in the hierarchy has the software, the software will be recursively copied down the hierarchy. Once the software is available, the agent will initiate the installation process of the software on the system. The above procedure is illustrated in detail referring to FIG. 5 which shows only a part of the entire IT network depicted in FIG. 1, [0085])*.

As to claims 7, 15, Mayer teaches the child update service node obtains the update payload for the software update from the parent update service node in a just-in-time fashion *(i.e. [0058] Configuration tools (CT) provide an abstract management interface on which the configuration and change management*

functionality of IT Configuration Management (ITCM) is based on. CTs provide all methods required to monitor and configure managed elements. Each category of managed element is associated with a configuration tool. Configuration tools support methods to manipulate elements (create, modify and delete) and to retrieve (query) elements of the associated type. CT's provide a protocol translation between an ITCM standard protocol and an element specific native protocol).

As to claims 8, 16, Melchione teaches the client update module distributes software updates to client computers according to rules established by an administrator via the administration API using the administration user interface *(i.e. In one case, administered software may be downloaded and run at a customer computer. In other cases, administered software represents some portion of administered software in processing communication with other administered software residing on other customer computers or at the provider. In another case, administered software may run at the provider with only a thin client (e.g., a web browser) running at a customer computer (e.g., displaying administered software output). All these cases, represent examples of provider-hosted application services, [0020]; In certain described examples, the software being administered is anti-virus software. New releases of anti-virus software can be automatically provided according to the software administration directives set by a customer administrator. In some case, administered software includes an agent that polls the data center to obtain administered software updates, [0021]).*

As per claim 9, Melchione teaches the root update service node further comprises a client update module for distributing software updates to client computers *(i.e. In one case, administered software may be downloaded and run at a customer computer. In other cases, administered software represents some portion of administered software in processing communication with other administered software residing on other customer computers or at the provider. In another case, administered software may run at the provider with only a thin client (e.g., a web browser) running at a customer computer (e.g., displaying administered software output). All these cases, represent examples of provider-hosted application services, [0020]; In certain described examples, the software being administered is anti-virus software. New releases of anti-virus software can be automatically provided according to the software administration directives set by a customer administrator. In some case, administered software includes an agent that polls the data center to obtain administered software updates, [0021])*.

Response to Arguments

Applicant's arguments filed 08/05/08 have been fully considered but they are not persuasive.

Applicant argues that:

(a) Claim 30, Melchione fails to teach an administration API "through which an administrator establishes a set of rules for distributing

software updates from the update service node to its child update service nodes."

(b) Claim 4, Mayer fails to teach "a reporting module for generating and sending update activity reports to the parent update service node."

(c) Claim 11, "an update service node comprising ... an administration application programming interface (API) through which an administrator establishes a set of rules for distributing software updates to its child update service nodes."

The Examiner respectfully disagrees for the following reasons:

Overview of the Applicants Invention to the teachings of Melchione, Mayer, Crudele.

A. The instant specification describes:

[0012] a method for **controlling the installation behaviors** of a computing device during a software update installation is presented. A software update for installation on the computing device is obtained. A determination is made as to whether an installation attribute is associated with the software update. If an installation attribute is associated with the software update, the installation behavior of the computing device in installing the software update is modified.

[0032] FIG. 1 software update system 100 is a block diagram illustrative of software update system 100 in accordance with the present invention. Generally described, the software update system 100 may comprise one or more client computing devices 110, an update service 120 and an external update provider 130. Generally described, the update service 120 stores and manages the **distribution of software updates** that are communicated to and installed on the client computing device 110. The software updates

may be provided by the update service 120 or by any number of external update providers 130.

B. The teaching of Mayer

Mayer similarly teaches:

*FIG. 9 is a block diagram showing an order management for **distributing software** within an IT environment, in accordance with the invention, Mayer, [0046].*

*FIG. 5 is an exemplary process for installing **software** on an IT resource connected to an IT network environment, in accordance with the invention, Mayer, [0042].*

*Before executing the associated changes, it is possible to verify the desired changes by database queries, e.g. for consistency checks. In case of problems, the change request can be reopened and modified according the needs. Once all the changes are verified, the automatic execution of the changes is started that provides **an automatic updating** of the respective resources of the managed IT environment. If the administrator is not satisfied at all it is possible to discard all changes, Mayer, [0084].*

C. The teaching of Melchione

Melchione analogously teaches:

*The technologies described herein can be used to administer software (e.g., one or more applications) across a set of administered devices via an application services provider scenario. Administration of software can include software installation, software configuration, software management, **software distribution**, software licensing functions, software acquisition, or some combination thereof. FIG. 2 shows an exemplary arrangement 200 whereby an application service provider provides services for administering software (e.g., administered software 212) across a set of administered*

devices 222. The administered devices 222 are sometimes called "nodes.", Melchione, [0050].

*At 922, **the software is packaged for distribution over a network. For example, software components and an installation program can be assembled into a package** (e.g., according to the CAB file specification of Microsoft Corporation), Melchione, [0094].*

*In certain described examples, the software being administered is anti-virus software. New releases of anti-virus software can be automatically provided according to the software administration directives set by a customer administrator. In some case, administered software includes an agent that polls the data center to obtain administered **software updates**, Melchione, [0021].*

D. The teaching of Crudele

Crudele similarly teaches:

*A software distribution system comprises building blocks including a preparation and test site, comprising a software package editor, software package transformation tools and an AutoPack module for preparing and testing software packages to be distributed to endpoints. A software package comprises a file including a complete definition of the actions involved in a software distribution and, once distributed, when a software package is available to a target endpoint, an engine resident on the target can be instructed via the management agent to decode the software package from the file into memory and then to perform various **software distribution operations including installing, removing and modifying the software and configuration of the endpoint**, Crudele, Abstract.*

*The system also able to support a wide set of reporting options including log files **updated at the end of any software** distribution operation and events triggered at the end of any software distribution operation, Crudele, col. 19, lines 39-42.*

The teachings of Melchione, Mayer, Crudele are directed the same filed to the Applicants as controlling distribution and installation updated software.

(a) Claim 30, Melchione does teach "an administration application programming interface (API) through which an administrator established a set of rules for distributing software updates from the update service node to its child update service nodes" as follows:

update service node equates to "The nodes can include agent software that periodically communicates with the data center 712" (Melchione, [0078]).

update service node to its child update service nodes equates to nodes 760A, 760B, 760C, 760D, and 760E being administered can be placed into one or more named logical groups 750A, 750B, and 750N of Melchione, [0077].

an administration application programming interface (API) equates to an interface of Melchione, [0059], *(i.e. an administrator is presented with **an interface** by which a policy can be applied to a group of devices, Melchione [0059])*.

a set of rules for distributing software updates equates to a policy of Melchione, [0059], *(i.e. Using an interface of this type, one or more directives can be bundled into a set of directives called a "policy." In the example, an administrator is presented with an interface by which **a policy** can be applied to a group of devices (e.g., a selected subset of the devices 222). In this way, **the administrator can control various administration functions (e.g.,***

installation, configuration, and management of the administered software *212) for the devices 222. In the example, the illustrated user interface 300 is presented in a web browser via an Internet connection to a data center (e.g., as shown in FIG. 2) via an HTTP-based protocol, Melchione [0059]; Activation of a graphical user interface element (e.g., element 312) can cause a request for application services to be sent. For example, application of* ***a policy to a group of devices may result in automated installation, configuration, or management of indicated software for the devices in the group****, Melchione, [0060]).*

a set of rules for distributing software updates equates to the configuration directives of Melchione, [0078] *(i.e. The nodes can include agent software that periodically communicates with the data center 712. During such communications,* ***the configuration directives stored in the database 726 can be implemented. For example, software administration commands, parameters, and software can be sent to an agent for use at a node****. Further details relating to configuration directives can be found in Melchione et al., U.S. Provisional Application No. 60/375,216, filed Apr. 23, 2002, entitled "Software Administration in an Application Provider Scenario Via Configuration Directives.", Melchione, [0078]).*

(b) Claim 4, Mayer does teach "a reporting module for generating and sending update activity reports to the parent update service node."

a reporting module equates to telling the agent that the software must be available and installed in the system of Mayer *(i.e. For example, **software is installed by telling the agent** that the software must be available and installed on the system. The agent will check whether the software is already installed on the system or whether it has to be installed. It will organize a copy of the software by asking its parent agent in the IT environment hierarchy to get the software. Once a node up in the hierarchy has the software, the software will be recursively copied down the hierarchy. Once the software is available, the agent will initiate the installation process of the software on the system. The above procedure is illustrated in detail referring to FIG. 5 which shows only a part of the entire IT network depicted in FIG. 1, [0085])*.

a reporting module equates to monitoring, reporting of Melchione *(i.esoftware administration can be accomplished via an application service provider scenario. Although **administration can include a wide variety of functions, the illustrated example enables monitoring** (e.g., for producing **reports** of virus infection), configuration, and installation of software. In addition, the polled pull scenarios described can allow the system to operate even though there may be a firewall in place. Thus, application administration can be performed in such a way that software is automatically updated through a firewall. Such an arrangement can provide a valuable service in many situations, such as for a large enterprise's information technology department. Such an enterprise may have 10, 100, 1000, 10,000, 100,000, or more nodes, Melchione, [0170])*.

a reporting module equates to *reporting options* of Crudele *(i.e. The system also able to support a wide set **of reporting options** including log files updated at the end of any software distribution operation and events triggered at the end of any software distribution operation, Crudele, col. 19, lines 39-42).*

(c) Claim 11, "an update service node comprising ... an administration application programming interface (API) through which an administrator establishes a set of rules for distributing software updates to its child update service nodes."

update service node equates to The nodes can include agent software that periodically communicates with the data center 712 of (Melchione, [0078]).

update service node to its child update service nodes equates to the nodes 760A, 760B, 760C, 760D, and 760E being administered can be placed into one or more named logical groups 750A, 750B, and 750N of Melchione, [0077].

an administration application programming interface (API) equates to an interface of Melchione, [0059], *(i.e. an administrator is presented with **an interface** by which a policy can be applied to a group of devices, Melchione [0059])*.

a set of rules for distributing software updates equates to a policy of Melchione, [0059], *(i.e. Using an interface of this type, one or more directives can be bundled into a set of directives called a "policy." In the example, an administrator is presented with an interface by which **a policy** can be applied to a*

group of devices (e.g., a selected subset of the devices 222). In this way, ***the administrator can control various administration functions (e.g., installation, configuration, and management of the administered software*** *212) for the devices 222. In the example, the illustrated user interface 300 is presented in a web browser via an Internet connection to a data center (e.g., as shown in FIG. 2) via an HTTP-based protocol, Melchione [0059]; Activation of a graphical user interface element (e.g., element 312) can cause a request for application services to be sent. For example, application of* ***a policy to a group of devices may result in automated installation, configuration, or management of indicated software for the devices in the group****, Melchione, [0060]).*

a set of rules for distributing software updates equates to the configuration directives of Melchione, [0078] *(i.e. The nodes can include agent software that periodically communicates with the data center 712. During such communications,* ***the configuration directives stored in the database 726 can be implemented. For example, software administration commands, parameters, and software can be sent to an agent for use at a node****. Further details relating to configuration directives can be found in Melchione et al., U.S. Provisional Application No. 60/375,216, filed Apr. 23, 2002, entitled "Software Administration in an Application Provider Scenario Via Configuration Directives.", Melchione, [0078]).*

Therefore, the prior arts, as combined, can not be distinguished from the

claim invention since these references read on the claimed limitations, as detailed.

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Miranda Le whose telephone number is (571) 272-4112. The examiner can normally be reached on Monday through Friday from 10:00 AM to 6:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, James K. Trujillo, can be reached on (571) 272-3677. The fax number to this Art Unit is (571)-273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (571) 272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Miranda Le/
Primary Examiner, Art Unit 2167